Institut de formation d'Ingénieurs de Paris Sud — **Année 2007-2008**

Recherche Opérationnelle

Correction du TD 2.

Recherche exhaustive

Exercice 1 :

Proposez un algorithme de recherche exhaustive pour le problème de sac à dos, et écrivez son pseudo code. Faites le "tourner" sur les données.

Pseudo-code :

```
//n = nombre d'objets, p = vecteur des poids, v = vecteur des valeurs
xmax = vecteur_nul(n)
vmax = 0
Pour i de 1 à n Faire
    x = convertir_en_vecteur(i,n) //Prend un nombre i et le transforme en vecteur binaire de longueur n
    Si p*x ≤ capacité Alors
        Si v*x > vmax Alors
            vmax = v*x
            xmax = x
        Fin Si
    Fin Si
Fin Pour
```

Pour le faire tourner, prenez un petit nombre d'objets auxquels vous donnerez un poids et une valeur aléatoire.

Proposez un algorithme de recherche exhaustive pour le problème des entrepôts, et écrivez son pseudo code.

Pseudo-code :

```
//m = nombre d'entrepôts, modèle = tableaux décrivants le PL
xmax = vecteur_nul(n)
ymax = vecteur_nul(m)
vmax = MAXINT
Pour i de 1 à 2^n Faire
    y = convertir_en_vecteur(i,m)
    [x,val,état] = résoudrePL(modèle,y)
    Si état == ok Alors
        Si val < vmax Alors
            vmax = val
            xmax = x
            ymax = y
        Fin Si
    Fin Si
Fin Pour
```

Algorithme glouton

Exercice 2 :

Proposez un algorithme glouton pour le problème des entrepôts, et écrivez son pseudo-code. Faites le "tourner" sur les données du TD 1, en représentant graphiquement les étapes et les flux.

Pour identifier les informations intéressantes, regardons sur l'exemple du projet comment nos variables évoluent. Premier cas : capacité infinie pour les entrepôts.

Comme la capacité est infinie, on sait qu'un seul entrepôt suffira pour satisfaire toute la demande. Il faut donc regarder :

- soit que son coût de construction soit le plus petit
- soit que les coûts de transports soient les plus faibles possibles
- soit les deux en même temps

Regardons ce qu'il se passe dans la troisième situation.
Comme les coûts indiqués correspondent aux coûts pour transporter toute la demande d'un client, on lit très simplement les coûts totaux :

	Brill	Hearthglen	Andorhal	Strahnbrad
Coûts fixes	200	250	300	350
Coûts de transports	1550	1425	1350	1450
Total	1750	1675	1650	1800

La solution est donc de construire un entrepôt à Andorhal, et de servir toutes les demandes à partir de cet entrepôt. Le coût est de 1650 pièces d'or. Notez que la solution n'est pas optimale.

Deuxième cas : la capacité des entrepôts est finie.
Ce qui nous intéresse ici c'est de savoir quels entrepôts sont construits, le coût total, et comment les demandes sont servies. On va introduire les demandes résiduelles pour suivre les demandes restant à satisfaire à chaque étape.
Notre algorithme va parcourir les entrepôts par ordre croissant de coût, et les ajouter un par un à chaque étape s'ils permettent de servir la demande. Pour chaque entrepôt choisi, on va alors regarder les coûts de transports unitaires, et satisfaire autant de demandes que possibles par ordre croissant de coût. On met à jour les demandes résiduelles pour savoir si on a obtenu une solution correcte.
Exemple sur les données "Warcraft" : capacité limitée à 800.

Étape 1 : initialisation
Entrepôts = {}
Coût = 0
X = []
Demandes restantes = [500 350 300 400 350 250]

Étape 2 : on ajoute Brill (le moins cher).
Entrepôts = {Brill}
Coût = 200
X = []
Demandes restantes = [500 350 300 400 350 250]

Étape 3 : on satisfait la demande de Lordaeron, qui est la moins coûteuse.
Entrepôts = {Brill}
Coût = 300

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 350 300 400 350 250]

Étape 4 : on satisfait la demande de Stratholme qui est la seconde moins coûteuse.
Entrepôts = {Brill}
Coût = 425

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 350 0 400 350 250]

Étape 5 : comme la capacité de Brill est épuisée, on ajoute l'entrepôt le moins cher suivant (Hearthglen).
Entrepôts = {Brill, Hearthglen}
Coût = 675

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 350 0 400 350 250]

Étape 6 : la demande la moins coûteuse est Mardenholde, on la satisfait.
Entrepôts = {Brill, Hearthglen}
Coût = 725

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 350 0 0 350 250]

Étape 7 : le demande suivante la moins coûteuse est Durnholde, qu'on satisfait.
Entrepôts = {Brill, Hearthglen}
Coût = 1025

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 350 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 350 0 0 0 250]

Étape 8 : la demande suivante est Dalaran, qu'on satisfait partiellement (capacité de l'entrepôt restante : 50)
Entrepôts = {Brill, Hearthglen}
Coût = 1089.29

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 350 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 300 0 0 0 200]

Étape 9 : on a épuisé la capacité de cet entrepôt, on ajoute le suivant.
Entrepôts = {Brill, Hearthglen, Andorhal}
Coût = 1389.29

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 350 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 300 0 0 0 200]

Étape 10 : les deux demandes restantes ont le même coût unitaire, on prend la première (Dalaran).
Entrepôts = {Brill, Hearthglen, Andorhal}

Coût = 1689.29

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 50 & 300 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 350 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Demandes restantes = [0 300 0 0 0 0]

Étape 11 : on prend la demande suivante qu'on satisfait.
Entrepôts = {Brill, Hearthglen, Andorhal}
Coût = 1939.29

$$X = \begin{bmatrix} 500 & 0 & 0 & 0 \\ 0 & 50 & 300 & 0 \\ 300 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 \\ 0 & 350 & 0 & 0 \\ 0 & 0 & 250 & 0 \end{bmatrix}$$

Demandes restantes = [0 0 0 0 0 0]

Étapes suivants : rien ne change jusqu'à ce qu'on ait testé tous les entrepôts.

Au final, on trouve un coût de 1939.29. Encore une fois, il ne s'agit pas de l'optimum. Le tableau X vous permet en outre de représenter les flux sortants de chaque entrepôt.

Un pseudo code correspondant serait quelque chose ressemblant à ceci :

```
// m = nombre d'entrepôts, n = nombre de clients
// capa_entr = capacité de chaque entrepôt
// pos_e donne la position du ième entrepôt le moins cher
// coutstransp = coûts de transports unitaires
// X : tableau des x_{i,j}, Y = vecteur des y_i
// cout(Y) calcule le cout de construction des entrepots
// transp(X) calcule les coûts de transport
pos_e = tri(couts_entr)// On trie les coûts, on n'a besoin que des indices.
Pour i de 1 à m Faire
    capacité = capa_entr(pos_e(i))
    capa_utilisée = 0
    // On va maintenant compléter les demandes
    p = colonne(coutstransp,i) //On récupère la colonne des coûts de transports à partir de l'entrepôt
    pos_p = tri(p) //On trie les coûts, on n'a besoin que des indices
    Pour j de 1 à n Faire
        //Est-ce qu'il reste de la demande à satisfaire?
        Si demande_restante(pos_p(j)) > 0
        //Si oui, on fait passer autant que possible
        X(pos_p(j),pos_e(i)) = min(capacité-capa_utilisée,demande_restante(pos_p(j)))
        //On met à jour la capacité utilisée
        capa_utilisée = capa_utilisée + X(pos_p(j),pos_e(i))
        Fin Si
    Fin Pour
    // Si la capacité utilisée est non nulle, alors c'est que l'on doit construire l'entrepôt
    Si capa_utilisée > 0
        Y(pos_e(i)) = 1
    Fin Si
Fin Pour
// On calcule le coût total
couttotal = cout(Y)+transp(X)
```